

Thoughts on Creating Better MMORPGs

By: Thomas Mainville

Paper 3: Tools and Methodologies

I. Introduction

Since MMORPGs are complex systems played simultaneously by a large base of widely distributed users, developers seeking to improve them must consider several areas of concern. These areas of concern range from optimization of message exchange between distributed components to more realistic modeling of real world systems in order to add to the complexity of the virtual world. In this paper I consider but a handful of the many technologies and methodologies currently being used or researched for use in MMORPGs. My goal in this paper is to consider technologies I view as potentially complementary in the design of a responsive, robust, and realistic online virtual world as a final step toward my final term paper where I will summarize and put together all of my ideas.

II. Tools

Matrix

Matrix is a distributed, low latency middleware system described by Balan et al. in (Balan et al. 391). It offers advantages over other methods for handling the numerous game clients that connect to typical MMORPGs. In (Balan et al. 390-391), Balan and his coauthors, prior to pointing out Matrix's advantages, first discuss some of the shortcomings of current MMORPG systems. They state that most MMORPGs "currently use a centralized server model, with players connecting to a single game server that handles the entire game world." This has the disadvantage of limiting the number of concurrent players a server can handle (30,000 according to (Balan et al. 390)). In order to handle players in excess of this limit, many games use "multiple servers that are statically assigned different parts of the game world" (Balan et al. 390). The authors point out that this technique can lead to a decrease in responsiveness when confronted with spikes in workload or heavy concentrations of players in particular areas (known as "hotspots").

Two techniques used to overcome this shortcoming are "overprovision[ing] the number of servers used for the game and/or impos[ing] artificial limits on the number of players that can be part of the map" (Balan et al. 390). The former proves to be very costly as extra computing resources must be purchased to ensure the ability to cope with peak volume. The latter can degrade the user experience by restricting the ability for players to enter certain areas of the game. (Note: I can attest personally to the frustration of not being able to play with friends because an area of the game is full.)

Rather than either of these approaches, Balan's team recommend the use of a "distributed system that can handle arbitrary game loads by dynamically and automatically adjusting

the number of servers used by the game in a scalable and efficient manner” (Balan et al. 390) . While advocating for this approach, Balan and colleagues point out that this system could be used on its own or in combination with static partitioning techniques, where it would step in to handle the aforementioned spikes in workload or the emergence of hotspots. Clearly, this technique, which is precisely the one used by Matrix, could be considered as conferring a degree of self-optimization or self-configuration to a MMORPG. In implementing this technique, however, care had to be taken to balance low message latency, which is necessary to guarantee a smooth, pleasant user experience with consistency, which is often sacrificed in the name of speed. Balan and his coauthors handle the apparently conflicting goals of low latency and high consistency by making a “key insight”. Namely, they observe that and MMORG is “an example of a nearly decomposable system...[in which] the number of interactions among subsystems, in some geometric space, is of a lower order of magnitude than the number of interactions within an individual subsystem” (Balan et al. 391).

The fact that this behavior “manifests itself though a “radius” or “zone of visibility associated with each game player” means each player need only be updated with details significant to his immediate scope, thus limiting the overhead of exchanging messages with parties for whom they hold no interest.. Matrix therefore provides “pockets of locally consistent state”. In a MMORPG, this could mean that if a character in a particular portion of the world was hunting goblins, only the goings on in his immediate area, such as the movement and actions of the goblins, and not the actions of a dragon on the other side of the virtual world. It’s clear to see the overhead savings realized by allowing PCs to be oblivious of that which does not concern them.

In a final word on Matrix, it should be noted that the developers make it easy to integrate into existing game frameworks. They provide an easy-to-use API that allows Matrix to be tied into an existing MMORPG with limited modification to that MMORPG. The creators of matrix describe this ease of integration in a list, shown here:

“No Change in Security Model: *A primary concern for online game developers is cheating and denial-of-service (DoS) attacks. In particular, they are quite resistant to any middleware that will lower their ability to tackle these issues. This concern naturally eliminates the use of peer-to-peer mechanisms, which fundamentally change the client-server interaction and security model. Matrix thus uses the same game developer preferred client-server architecture, as shown in Section 3, allowing the developer to reuse existing anti-cheating and anti-DoS mechanisms.*

Separation of Concerns: *To make developing distributed games easier, Matrix provides a clean “separation of concerns” programming model where Matrix would handle the distributed computing aspects of a game such as consistency, scalability, resource provisioning and fault-tolerance, leaving the MMOG developer to focus on the core game logic.*

Support Multiple Gaming Platforms: *Game developers frequently develop games for multiple gaming platforms; having to write new Matrix routines for each platform would*

hinder adoption. Our APIs do not require any new Matrix-specific routines for a new platform.

Simplicity: *Building and debugging a large distributed system is a tricky endeavour. As such, Matrix intentionally uses the simplest possible algorithms and APIs. The simple algorithms allow Matrix to be easier to debug and maintain, and the API allows existing games to be quickly and easily modified for use with Matrix.”*

NeL

If there is any doubt as to the competitive nature of the MMORPG market, the fate of developers like Nevrax should do much to prove otherwise. Nevrax, the French MMORPG developer of a game known as Ryzom ended up in receivership (Nevrax, 1), though Ryzom apparently is still supported (see: www.ryzom.com). Prior to meeting this fate, however Nevrax made a valuable contribution to the MMORPG development community with the release to the open source community of their game engine, known as NeL (for Nevrax Library). In (Wen, Nel 1), the author Wen describes NeL as: “a toolkit for the creating 3D-graphic MMORPGs or similar online game-play environments that require both client and server code. It runs on the Linux and Windows OSes, using OpenGL as its 3D graphics renderer” (Wen, NeL1).

NeL was developed with three goals in mind:

- “Immersion: NeL must be able to render high-quality 3D graphics.”
- “Massive: the engine must maintain a good level of performance while displaying a large online population of players and other characters.”
- “Open: the engine code must be easily understandable and extendable for the benefit of the open source and free software community.”

These goals are evident when one considers the library’s features. NeL provides not only the 3-D graphics engine necessary for development of a MMORPG, but also a number of other features, including networking capability, AI (of particular interest to me), and facilities for in-game audio. The graphics engine apparently supports a great deal of customization, allowing “flexibility for modeling outdoor settings, with better rendering, and lighting model capabilities” (Wen, NeL 1). Clearly NeL presents a potentially intriguing tool for researchers desiring to test concepts (such as the application of biologically inspired ideas) in a “home grown” MMORPG setting. Though the work to get a viable MMORPG up and running using NeL might be daunting, the fact that commercial developers are likely not too keen to release their source code to researchers makes it worth consideration.

Worldforge

An alternative to Nel is another non-commercial product known as Worldforge. Described in (Wen, WorldForge 1), Worldforge is less feature rich than NeL as noted in (Wen, NeL 1), but seems appropriate for the development of smaller-scale projects. I

envision such projects as including small, simple worlds for the testing of single self-programmed subsystems. This vision seems to be supported when one considers the two projects developed on the Worldforge platform to date, Acorn and Mason. These two games are very limited in scope, offering no threat to players from NPCs or PCs. My concept of Worldforge is further backed up by Wen in (Wen, WoldForge 1) when he mentions that Worldforge might be appropriate for “the type of online games which might not be viable in the commercial marketplace”. To summarize, what Worldforge lacks in feature richness and robustness, it may make up for in simplicity of use and appropriateness for simple research applications.

NetLogo

Netlogo requires no introduction. The initial description in the NetLogo manual sums it up nicely: “NetLogo is a programmable modeling environment for simulating natural and social phenomena. It was authored by Uri Wilensky in 1999 and is in continuous development at the Center for Connected Learning and Computer-Based Modeling.” (NetLogo User Manual 1) A popular choice for modeling any number of processes with behavior that emerges over time, Netlogo seems to present some interesting possibilities for modeling complex MMORPG subsystems. Even if it can’t be applied directly to a MMORPG (as an actual part of the game system), NetLogo has the potential to provide valuable predictive analytical data regarding how things will or have unfolded in a complex virtual world.

Specifically, I see NetLogo being used as a valuable tool for a number of purposes, including, among other things:

1. The prediction of patterns of agents in intelligent MMORPGs before those agents are deployed into the actual virtual world (e.g. modeling NPC populations, NPC interactions, NPC behaviors, etc.)
2. The analysis and/or troubleshooting of unforeseen complications brought on by unpredictable behavior in MMORPGs (e.g. goblins running amok, financial meltdown, migration patterns, etc.).
3. Aiding in the development of efficient and correct feedback loops as mentioned in my second paper.
4. Simulating PC actions in the aggregate (assuming agents could be written to reasonably approximate PC actions)

MPML3D

MPML3D, which stands for Multimodal Presentation Markup Language 3D (MPML3D), is a markup language that can be used to control behavior of NPCs in a 3D virtual world. Ullrich and his colleagues describe their interesting use of MPML3D to simulate NPCs in the MMOG Second Life in (Ullrich et al. 281-287). I choose the word “simulate” here

because this game does not actually support NPCs. Ullrich et al. are, in fact, using MPML3D to script the control avatars (which would normally be player-controlled PCs) within the Second Life environment. I found the work of these researchers interesting because of its potential applicability to other MMOGs, including MMORPGs. Specifically, it is easy to imagine such a technique for PC control being used in a “home grown” MMORPG environment developed by a research team using NeL or Worldforge. NPC behavior could be scripted in MPML3D, and the logic behind the scripts could be based on information gleaned from simulation using a tool like NetLogo. The result of such a use would be NPCs exhibiting potentially very intelligent behavior in a carefully crafted game world

III. Conclusion

There are a number of tools and methodologies that show great promise for application in the MMORPG research. Some are suitable for integration into the code of a virtual world, while others are best used in a support role. To illustrate the potential use of the small number of tools discussed in this paper, I will conclude with a brief proposal for their combined application.

In a theoretical game development process where these tools are applied:

1. The world and its subsystems are prototyped using a “light weight” tool such as Worldforge.
2. At the same time, NetLogo (or a similar modeling tool) is used to simulate the various subsystems, which are then applied in the Worldforge model to verify their validity.
3. PCs are introduced to the world in high numbers with their behavior defined by markup such as MPML3D, which is created in response to the findings from NetLogo simulations.
4. Once the world and its subsystems are reasonably fleshed out, they are ported to a more robust MMORPG platform like NeL
5. Matrix is integrated into this ported version of the MMORPG, thereby ensuring an optimal balance between performance and consistency.
6. The MPML3D used for governing NPC control from 3 is refined given the increased level of complexity made possible by the more robust platform.
7. The foundations of an expansive, engaging MMORPG virtual world have been laid.

Please note that the above is but a suggested process for development based on my very superficial understanding of each proposed tool. While I believe the proposed process to be fairly sound, the appropriateness of the proposed tools warrants further investigation.

References

Balan, Rajesh Krishna et al. Matrix: Adaptive Middleware for Distributed Multiplayer Games. Ed. G. Alonso. IFIP International Federation for Information Processing. 2005.

Nevrax goes into receivership. Eurogamer (via gameindustry.biz). 2006. <<http://www.eurogamer.net>>.

Wen, Howard. NeL: The Software Behind the Next Great MMORPG?. O'Reilly Linux Devcenter. 2003. <<http://www.linuxdevcenter.com>>.

Wen, Howard. WoldForge: In Pursuit of Open Source, Massive, Online Games. O'Reilly Linux Devcenter. 2002. <<http://www.linuxdevcenter.com>>.

NetLogo 3.1.4 User Manual

Ullrich, Sebastian, et al. Extending MPML3D to Second Life. Ed. H. Prendinger, J. Lester, and M. Ishizuka.